



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №6
Архітектура комп'ютера
«СИСТЕМА ОБРОБКИ ПЕРЕРИВАНЬ АРХІТЕКТУРИ IA-32 (X86) У
REAL ADDRESS MODE»

Виконали:
студенти групи ІТ-01:

Перевірив:
Бердник Ю. М.

Бардін В.Д.
Задніпрянець А.А.
Куркін О.О.

Дата здачі 22.04.2021

Захищено з балом _____

Київ 2021

Вихідний код:

```
1 TITLE ЛР_6
2 ;-----
3 ; Дисципліна: Архітектура комп'ютера
4 ; НТУУ "КПІ"
5 ; Факультет: ФІОТ
6 ; Курс: 1
7 ; Група: IT-01
8 ;-----
9 ; Автор: Бардін, Задніпрянець, Куркін
10 ; Дата: 07/04/2021
11 ;-----
12 IDEAL
13
14 ;-----
15 MACRO M_Exit ; Exit macros
16 ; Input: AL = Exit code
17 ; Output: ---
18 mov ah, 04Ch ; DOS interruption number to exit
19 int 21h
20 ENDM
21
22 ;-----
23 MACRO M_Init ; Initiate DS and ES
24 mov ax, @data ; ax <- @data
25 mov ds, ax ; ds <- ax
26 mov es, ax ; es <- ax
27 ENDM
28
29 MODEL small
30 STACK 256
31
32 DATASEG
33 msg db "Bardin, Zadnipyranets, Kurkin $"
34
35 bak_int0Bh_offset DW ? ; Effective procedure address - default handler for COM1 interrupt
36 bak_int0Bh_seg DW ? ; Segment begin address for hardware conversion function COM1
37
38 CODESEG
39
40 PROC main
41 M_Init
42
43 ; Stage 1. Receiving effective address and interrupt offset-----
44 mov di, 61h
45 call get_int_vector
46 ; Output arguments: (bx - effective address, es - segment address )
47 mov [bak_int0Bh_offset], bx
48 mov [bak_int0Bh_seg], es
49
50 ; Stage 2. Збереження стандартного обробника переривань COM 2 за іншим вектором
51 mov di, 61h ; Input parameter for set_int_vector, new interruption number
52 mov dx, bx ; Offset for procedure, effective address
53 call set_int_vector
54
55 ; Stage 3. Set new handler for custom interruption
56 mov di, 61h
57 mov dx, OFFSET int61h
58 mov ax, SEG int61h
59 mov es, ax
60 call set_int_vector
61
62 int 61h ; call interruption
63
64 ; Stage 4. Enroll all changes
65 mov di, 61h
66 mov dx, [bak_int0Bh_offset]
67 mov ax, [bak_int0Bh_seg]
68 mov es, ax
69 call set_int_vector
70
71 xor al, al ; Set al to zero
72 M_Exit
73 ENDP main
```

```

74 ;-----
75 PROC get_int_vector
76 ; Purpose: Get logical address for procedure by interruption vector number
77 ; Input: DI <- Interruption vector number
78 ; Bxixd: BX <- procedure's effective address
79 ; ES <- procedure's segment address
80 ;-----
81 ; Add registers to stack
82 push ax
83 push di
84
85 xor ax, ax ; Clean AX
86 mov es, ax ; Move to 0000h
87 shl di, 2 ; Left logical shift, same as multiply to 4
88 mov bx, [es:di] ; Write handler's effective address to BX
89 mov ax, [es:di + 2] ; Write handler's segment address to AX
90 mov es, ax ; Write handler's segment address to ES
91 ; Restore registers
92 pop di
93 pop ax
94 ret ; Return from procedure
95 ENDP get_int_vector
96
97 ;-----
98 PROC set_int_vector
99 ; Purpose: Set new a handler to interruption number
100 ; Input: DI - interruption number, where the procedure will be bind
101 ; DX - new handler effective address
102 ; ES - new handler segment address
103 ; Output: ---
104 ;-----
105 cli ; Disable hardware interrupts
106
107 ; Adding registers to stack
108 push ax
109 push di
110 push ds
111
112 xor ax, ax ; Clean AX
113 mov ds, ax ; Move to 0000h
114 shl di, 2 ; left logical shift, same as multiply to 4
115 mov [ds:di], dx ; Write effective address to the first half of vector
116 mov [ds:di + 2], es ; Write segment address to the second half of vector
117
118 ; Restore registers
119 pop ds
120 pop di
121 pop ax
122 sti ; Enable hardware interrupts
123 ret
124 ENDP set_int_vector
125
126 ;-----
127 PROC int61h
128 ; Purpose: New interruption (61h)
129 ; Input: ---
130 ; Output: ---
131 ;-----
132 mov ah, 09h ; print a message
133 mov dx, offset msg
134 int 21h
135
136 mov al, 20h ; exit to operation system
137 out 20h, al
138 iret ; return from interruption
139 ENDP int61h
140 END main ; Кінець коду

```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Z:\>d:
D:\>set path=c:\tasm
D:\>tasm /zi L6.ASM >X:\ASM.LOG

Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:    L6.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  464k

D:\>if exist L6.OBJ tlink /v/3 L6.obj >X:\LINK.LOG

Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\>L6.exe
Bardin, Zadniprovanets, Kurkin
(END)Here is the end of the program's output

Do you need to keep the DOSBox [Y,N]?
```

Результати дослідження:

Проаналізувавши отриманий код, можна дійти висновку, що переривання це звичайні процедури, які фактично є глобальними й доступні не в контексті однієї програми, а для всіх програм у системі. Також ми створили власне переривання, що реалізує потрібну процедуру.

Висновок: В ході виконання лабораторної роботи, ми розробили власне програмне переривання. Дослідили передумови для створення власного переривання. По-перше, для гнучкості розробки і зменшення обсягу коду. По-друге, для реалізації функції обробки нового апаратного переривання для управління нестандартним обладнанням МПС. По-третє, для вдосконалення або повного переписання процедури обробки переривання, що запрограмована у ОС.